Design, Implementation and Performance of an Edge Computing Prototype Using Raspberry Pis

Tyler Gizinski
School of Computing
Grand Valley State University
Allendale, Michigan, USA
gizinskt@mail.gvsu.edu

Xiang Cao
School of Computing
Grand Valley State University
Allendale, Michigan, USA
caox@gvsu.edu

2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) | 978-1-6654-8303-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCWC54503.2022.9720848

***Abstract*—Edge computing nowadays has been a novel computing paradigm, which intends to improve the performance of many applications. In edge computing, edge servers are usually deployed at locations that are closer to data sources, so that latency and response time can be reduced. Recently, Raspberry Pi has become a popular small low-cost computer that can be deployed for many Internet of Things (IoT) applications.**

In this paper, we design and implement an edge computing prototype using Raspberry Pi devices. We demonstrate the feasibility of edge computing with Raspberry Pis as edge servers. We take advantage of the local and parallel processing capabilities of Raspberry Pis to show the scenarios where edge computing can outperform the traditional cloud computing. Hands-on experiments verify that in some cases, edge computing has better performance in terms of total finishing time of applications.

***Index Terms*—Edge Computing, Prototype, Raspberry Pi, Design, Implementation, Performance**

I. Introduction

Edge computing [1] [2] [3] [4] has been prevalent in recent years. As a novel computing paradigm, edge computing can provide many data processing services at locations that are closer to the end users. In that case, the Quality of Service (QoS) for users can be improved, as the latency and response time from servers are reduced. Security, privacy and other issues can also be enhanced in edge computing.

In today's Internet of Things (IoT) era, many IoT devices are generating and/or collecting huge amounts of data. In traditional IoT applications, due to the low processing and storage capabilities of IoT devices, these data are sent to the Cloud for further actions. However, in edge computing, as the intermediary tier between the Cloud and end IoT devices, edge servers can help store and process data, so that not all of the data need to be transmitted to the Cloud.

Recently, Raspberry Pi has been developed as a popular small low-cost computer for many IoT applications. Raspberry Pi nowadays is not only a simple device for data collection, but also can process and store data as servers. In this paper, we use Raspberry Pi devices to build a proof-of-concept prototype to demonstrate the feasibility of edge computing and show the performance compared with traditional cloud computing. Particularly, we conduct hands-on experiments and investigate the performance in terms of the total finishing time of applications including data processing time and data transmission time.

In our edge computing prototype, we believe the Raspberry Pi devices can work as edge servers for data processing and storage, as they have more and more powerful CPUs, memories and storage capacities. If Raspberry Pi devices only work as data collectors without any further processing and send every data to the Cloud, much computing power is wasted. Hence, our prototype not only intends to reduce the data transmission time by taking advantage of Raspberry Pi's local processing capability, but also it effectively utilizes the otherwise unused distributed computing power. In addition, we take advantage of the parallel processing capability offered by multiple Raspberry Pis as affordable edge devices.

The contributions of our paper are as follows.

- We build an edge computing prototype and use Raspberry Pi devices as edge servers for data processing and storage.
- We take advantage of Raspberry Pi's local processing capability to reduce the data transmission time of applications, and harness the otherwise unused distributed computing power of Raspberry Pi devices. We also utilize the parallel processing feature provided by multiple edge devices.
- We demonstrate that edge computing is feasible and promising, by conducting hands-on experiments and investigating the performance in terms of the total finishing time of applications.
- We discuss the scenarios where edge computing outperforms traditional cloud computing in our prototype setting.

In this paper, we believe our prototype and the results of the experiments can be examples to show the edge computing performance in terms of the total finishing time of applications. We also hope this paper can contribute to the community for helping people better understand the advantages and limitations of edge computing paradigm.

The rest of the paper is organized as follows. Section II is related work. We show the background in Section III. In

978-1-6654-8303-2/22/$31.00 ©2022 IEEE

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.

Section IV, our prototype architecture is presented. The design and implementation are discussed in Section V. In Section VI, we demonstrate the performance evaluation. We finally conclude the paper and discuss the future work in Section VII.

II. Related Work

Research articles in [1] [2] [3] [4] focused on the general overview of edge computing. They discussed the key aspects, and provided surveys of edge computing systems. Advantages, visions and challenges of edge computing were also presented in these work.

There are some research work about specific edge computing prototypes and systems. In [5], the authors presented an edge computing system for virtual being creation along with the implementations. In [6], a novel architecture system integrating edge computing into low earth orbit satellite networks was proposed to improve the system performance compared with cloud computing. In [7], the authors presented a virtual edge computing prototype for vehicle communications and used the vehicular micro cloud as a virtual edge server. In [8], the authors applied the concept of cloud, fog and edge computing to demonstrate an enterprise architecture of a smart airfield lighting system. Research work in [9] was conducted for a real-time surveillance system using the edge computing architecture. The authors also implemented the prototype of the system, evaluated the performance, and demonstrated the effectiveness and feasibility.

Some existing work in [10] [11] [12] [13] are related to edge computing research using Raspberry Pis. In [10], the authors presented an edge computing hardware prototype solving a distributed optimal power flow to coordinate agents of distributed energy resources in a low-voltage electrical network. Raspberry Pis were used as prosumer agents. In [11], the authors investigated how to use a Raspberry Pi cluster for deploying part of an orchestration platform. The objective of the platform was for more complicated data collection and analysis in the edge of a cloud. Research work in [12] demonstrated an edge cloud PaaS architecture and showed that the edge cloud requirements could be met by using Raspberry Pis to implement container and cluster technology. In [13], the authors discussed the performance of machine learning in tourism applications on a Raspberry Pi cluster as an edge server using Hadoop and Spark techniques.

There are some other interesting research work based on Raspberry Pi implementation. In [14], the authors implemented Hadoop MapReduce framework in a Raspberry Pi cluster to efficiently access and manage data from sensor nodes. In [15], the authors presented a smart home automation system consisting of Raspberry Pis. They integrated different sensors together into a Raspberry Pi and connected each major appliance using a web server. In [16], the authors described a wireless sensor network system using affordable Raspberry Pis and Arduino devices for environmental monitoring applications. Research work in [17] presented a low-cost energy-efficient cluster consisting of Raspberry Pis to save power consumption for data mining algorithms. In [18], the authors discussed experiences of replicating software performance experiments using Raspberry Pis and showed good replication results. In [19], the authors used MPI on a Raspberry Pi cluster for parallel and distributed computing tests and demonstrated the potential of the cluster. In [20], a cluster consisting of 64 Raspberry Pi nodes was developed. The authors presented the benchmark results of the cluster including computational power and network performance.

III. Background

A. Total Finishing Time of Applications

In many traditional IoT applications, IoT devices first collect data. Sometimes, IoT devices can process the data and send the results/conclusions of data to the Cloud. However, due to limited computing and storage capacities of IoT devices, usually data are directly sent to the Cloud for processing to generate the results/conclusions. In both cases, the results/conclusions of data are eventually acquired by the Cloud.

From the perspective of Quality of Service (QoS), the total finishing time of applications is an important performance metric. Here in this paper, we define the total finishing time as the sum of data processing time and the data transmission time (i.e., the time on the network from IoT devices to the Cloud). In other words, the total finishing time is the elapsed time from the moment after data are collected, to the moment when the Cloud obtains the results/conclusions. It is always better to reduce the total finishing time so that the results/conclusions of data can be available in the Cloud as quickly as possible.

B. Edge Servers and Raspberry Pi

In edge computing, edge servers become a new tier that resides between the Cloud and end devices. In IoT applications, edge servers can process data at locations that are closer to the IoT devices, and send the results/conclusions to the Cloud.

In this paper, we believe Raspberry Pis are good devices to emulate edge servers for the edge computing paradigm. Compared with regular computers, Raspberry Pis have less powerful processing capability, similar to edge servers vs. the Cloud. In many IoT applications, Raspberry Pis are deployed at locations that are very close to, or even at the same places as data sources (Data sources, such as cameras, sensors, etc. are often attached to Raspberry Pis). This fact is also similar to the feature of edge servers.

Due to their small sizes, Raspberry Pis are very portable and can be easily deployed in many places as distributed computing devices. We believe Raspberry Pis are valuable computing resources that should be effectively utilized. Hence, in this paper, we also intend to process data in Raspberry Pis to harness otherwise unused distributed computing power.

C. Our Perspective

Since edge servers usually have less processing power than the Cloud, in this paper, our perspective is to reduce the

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.

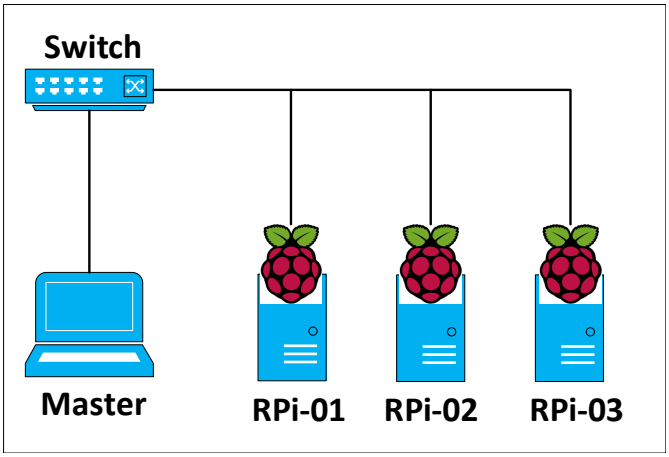

Fig. 1. Prototype Architecture

total finishing time of applications by decreasing the data transmission time using the idea of edge computing.

In order to reduce the data transmission time, we intend to decrease the data amount sent from edge servers to the Cloud. Specifically, in our edge computing prototype, the Raspberry Pis process data and only send the results/conclusions of the data to the computer as the Cloud. In that case, although the data processing time is increased due to the hardware limit of Raspberry Pis, we believe at least in some scenarios, the reduction of data transmission time could compensate and still could decrease the total finishing time.

We also consider the potential parallel processing feature provided by edge computing. As an edge server is usually more affordable than a Cloud server, we believe parallel processing offered by multiple edge servers is very promising and can lead to better performance in some cases.

In this paper, we compare the above strategy inspired by edge computing with the method of traditional cloud computing (where Raspberry Pis do not process data, but send all the raw data to the Cloud), and discuss the performance.

IV. Prototype Architecture

Our edge computing prototype consists of a master computer and three Raspberry Pis, connected by an Ethernet switch, as shown in Figure 1. Three Raspberry Pis serve as edge servers. The master computer is an emulation of a Cloud server, which can process data faster than the edge servers (Raspberry Pis).

In our edge computing scenario, Raspberry Pis process data and send the results/conclusions to the master computer. In that case, the master computer does not process data.

To fairly compare the performance with the cloud computing, we use the same architecture and devices. In the cloud computing scenario, the Raspberry Pis send raw data directly to the master computer without processing them. After receiving the data, the master computer processes them to obtain the results/conclusions.

V. Design and Implementation

A. Applications

In this paper, we intend to show the benefit of edge computing. As discussed above, the goal of our edge computing prototype using Raspberry Pis is to sacrifice data processing time for data transmission time. Hence, when we select applications, we have two criteria as follows.

- The input data size of the application is large.
- The output data (result/conclusion) size of the application is small.

As mentioned above, in our edge computing prototype, the Raspberry Pis process input data first and send the output data (results/conclusions) to the Cloud, whereas in the comparison method of cloud computing, Raspberry Pis directly send input data to the Cloud, which processes them to generate the output.

Hence, our edge computing prototype avoids sending a large amount of input data by only transmitting a small-size output data (result/conclusion) to the Cloud, which greatly reduces the data transmission time. However, in the cloud computing paradigm, it takes a long time to transmit a large-size input data from Raspberry Pis to the Cloud.

In this paper, we select and implement two applications as follows.

1) Smith-Waterman Gene Sequence Alignment [21] [22]:

The first application is Smith-Waterman gene sequence alignment. Gene sequence alignment requires two input files containing gene strings and returns the length of a matched sequence between the two strings. In Smith-Waterman algorithm, a matrix is constructed and a similarity score is generated for each character in the sequence. Then a traceback algorithm looks for the path with the greatest similarity (highest score).

In this application, the input data are two gene character strings stored in input files. The problem input size becomes the product of the sizes of two input gene character strings. We use Smith-Waterman algorithm to generate the similarity score, which represents the similarity between the two character strings, as the output (result/conclusion).

Hence, we can see that this application follows our above criteria, as the output data (result/conclusion) is very small (only a number), whereas the input data are very long character strings, which are much larger.

2) Tesseract for Optical Character Recognition [23]:

The second application is called Tesseract, an Optical Character Recognition (OCR) program. Tesseract for OCR requires an image as the input data and returns a string containing the characters recognized from the image file as the output (result/conclusion).

Thus, we can find that this Tesseract application also follows our criteria. The output data (result/conclusion) is only a string which usually consists of several/some characters in Bytes, whereas the input data is an image whose size is at least in KBs or MBs.

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.

B. Connections

In our implementation, to communicate between Raspberry Pis and the master computer, we use simple socket connections and the FTP protocol based on different scenarios.

1) Socket Connections:

Straightforward socket connections are used when simple requests (e.g., requests for data, initialization) are transmitted between Raspberry Pis and the master computer. Also, Raspberry Pis use socket connections to send the output data (results/conclusions) to the master computer due to their small sizes. In other words, straightforward socket connections are for transmitting small-size messages.

2) FTP Protocol:

When Raspberry Pis send the input data (i.e., gene sequence files, or images) to the master computer in the cloud computing scenario, the FTP protocol is used. Due to the large data sizes, it is ideal to adopt a dedicated protocol to transfer files. In our experiments, we also compare FTP with the SCP protocol for file transfer, and find that FTP always outperforms SCP in terms of transmission speed. Hence, in this paper, FTP is used for all the data transmission of gene sequence files and images in the cloud computing scenario.

C. Edge Computing vs. Cloud Computing

In this paper, we design two edge computing modes and one cloud computing mode for experiments. In edge computing modes, as mentioned previously, the Raspberry Pis process input data and send the output (results/conclusions) to the master computer. On the contrary, in cloud computing mode, the Raspberry Pis do not process the input data, but send them to the master computer for processing. The details of these modes are as follows.

1) Edge Computing - Individual Mode:

In this mode, the master computer first sends requests to Raspberry Pis, asking for results/conclusions of input data. After receiving these requests, the Raspberry Pis process input data and transmit the outputs (results/conclusions) to the master computer.

Specifically, in this individual mode, each time a Raspberry Pi finishes processing an input data (e.g., an image, or two gene sequence files), an output message is sent to the master computer. This means if 100 images or 100 pairs of gene sequence files are processed, 100 separate individual messages of outputs are sent. Each message contains the result of only an input data.

2) Edge Computing - Compact Mode:

The compact mode is similar to the individual mode generally. The difference is that, in this compact mode, each time a Raspberry Pi finishes processing an input data, the output is temporarily stored instead of being immediately sent out. After all the input data are processed, multiple results are packed together in a single message separated by semicolons (e.g., "3;3;3;3;...3;3;3", assuming '3' is the output of an input data). Then this single message (instead of multiple ones) is sent to the master computer. The compact mode is an attempt to even further reduce network traffic and the master computer's activity.

In both individual and compact modes, not only less data are transmitted between the master computer and Raspberry Pis to reduce the data transmission time, but also the system takes advantage of parallel processing of input data on multiple Raspberry Pi devices.

3) Cloud Computing:

In the cloud computing scenario, the master computer first sends requests to Raspberry Pis for initial input data. Then Raspberry Pis transmit the input data to the master computer, which further processes them to obtain the outputs (results/conclusions).

In all the above scenarios, the total finishing time of applications in our implementation is defined as the elapsed time from the moment when the master computer sends out the requests, to the moment when either the master computer receives the output from Raspberry Pis (edge computing scenario), or the master computer obtains the output after it processes the input data from Raspberry Pis (cloud computing scenario).

VI. Performance Evaluation

A. Experiment Setup

In this paper, we use a Gateway Netbook (Atom N455 CPU @ 1.66GHz, 512 KB cache, 2 GB RAM, OS: Ubuntu 18.04.5 LTS) as the master computer. Three Raspberry Pis (Model B Rev 2, ARMv6 CPU @ 700 MHz, 128KB cache, 512MB RAM, OS: Raspbian GNU/Linux 10) run as edge servers. We use the total finishing time of applications as the performance metric. The data processing time and data transmission time are also shown respectively in our results.

B. Experiments Conducted

We conduct experiments for both Smith-Waterman (Gene Sequence Similarity Score) and Tesseract (OCR) applications using three different input data sizes (small, medium, and large) on multiple numbers of Raspberry Pis (1, 2, and 3) respectively, including two edge computing modes and the cloud computing scenario.

In each set of experiments, we intentionally iterate executions for data (i.e., repeat processing the same data multiple times) to make the input data large enough and to show the difference between two edge computing modes. Also, in the cloud computing scenario, the quantity of executions is increased at a folding rate to correlate back to the number of Raspberry Pis. For example, when two Raspberry Pis are used and each one repeats processing an image 25 times in edge computing modes, then in the cloud computing scenario for a fair comparison, the master computer iterates executions for the same image for 50 times.

In this paper, the number of iterations for processing input data by a Raspberry Pi in edge computing modes are 100 times (Smith-Waterman) and 25 times (Tesseract) respectively, as shown in the results below.

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SMITH-WATERMAN, SMALL-SIZE, 1 RPI, CLOUD: 100 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	0.0652	0.1464	0.2116
Edge - Compact	0.0553	0.0019	0.0572
Cloud	0.0330	36.7222	36.7552

TABLE II
SMITH-WATERMAN, SMALL-SIZE, 2 RPIS, CLOUD: 200 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	0.0934	0.7372	0.8306
Edge - Compact	0.0774	0.0051	0.0825
Cloud	0.0690	69.4561	69.5251

TABLE III
SMITH-WATERMAN, SMALL-SIZE, 3 RPIS, CLOUD: 300 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	0.0657	0.1433	0.2090
Edge - Compact	0.0580	0.0017	0.0597
Cloud	0.0982	110.7317	110.8299

C. Results, Projections and Discussion

1) Smith-Waterman for Similarity Score of Genes:

- Small-size Input:

  In this set of experiments, we manually create two input files, which contain gene sequences "GATATGCA" and "ATAGCT" respectively. The sizes of two input files are only 8 Bytes and 6 Bytes, so that the problem input size is 48 (8*6) Bytes. We show the results of experiments using 1, 2 and 3 Raspberry Pis respectively in Tables I, II and III. We can see that data processing takes little time to finish due to the small input size. In the cloud computing, entire data need to be transmitted to the Cloud for processing, whereas in the edge computing, only the results/conclusions (whose sizes are very small) are sent. Hence, in all tables, both individual and compact modes in edge computing outperform the cloud computing scenario in terms of the total finishing time, as the dominant time for the cloud computing scenario is the data transmission time. The compact mode of edge computing indeed significantly reduces the data transmission time compared with the individual mode, as fewer messages are sent.

  Due to the budget limit, we do not have more Raspberry Pis to further show the performance comparison of edge computing in a larger scale and the cloud computing. Hence, we estimate the performance assuming more than three Raspberry Pis are deployed. To have a fair and more convincing comparison, we intentionally compare the performance of cloud computing with the individual mode of edge computing, which takes more time to finish than the compact mode. We use the average total finishing times in Tables I, II and III as the data for the projected performance.

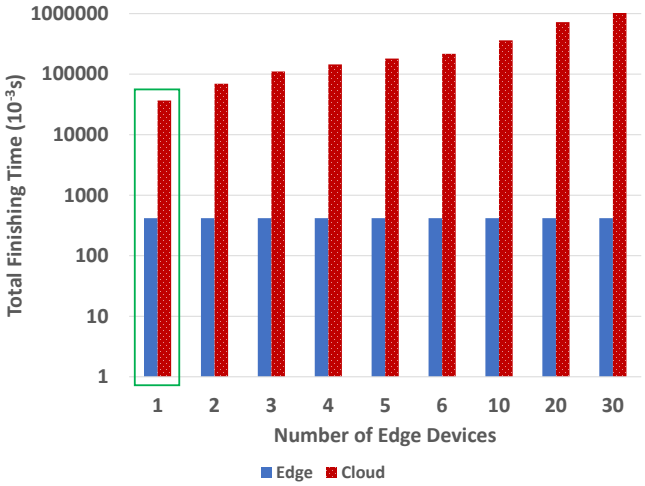

Fig. 2. Projected Performance: Smith-Waterman, Small-size Input

  Figure 2 shows the projected performance in logarithmic scale. We can see that with more Raspberry Pis, the edge computing maintains the same total finishing time, as these devices can process and transmit data in parallel. However, the total finishing time of cloud computing grows as the number of iterations increases (due to the increase of edge devices for a fair comparison mentioned in Sec VI(B)), so that the Cloud has to deal with data sequentially. We also find out that in this small-size input scenario, edge computing always outperforms the cloud computing (with only 1 Raspberry Pi, the edge computing already spends less time than the cloud computing). This is because with small-size input, the dominant component of total finishing time is data transmission time in the cloud computing, and the edge device does not need to spend much time to process data in the edge computing scenario. Hence, the advantage of reduced-size and faster data transmission in edge computing can be shown immediately even with only 1 Raspberry Pi.

- Medium-size Input:

  In this set of experiments, the two input files contain gene sequences selected from resources in [24] [25] respectively. The sizes of two input files are 9382 Bytes and 48 Bytes, so that the problem input size is about 450KB (9382*48). Tables IV, V and VI demonstrate

TABLE IV
SMITH-WATERMAN, MEDIUM-SIZE, 1 RPI, CLOUD: 100 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	74.9473	0.1725	75.1198
Edge - Compact	71.9403	0.0019	71.9422
Cloud	31.8010	35.6732	67.4742

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.

TABLE V
SMITH-WATERMAN, MEDIUM-SIZE, 2 RPIS, CLOUD: 200 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	83.8867	0.2305	84.1172
Edge - Compact	73.9825	0.0076	73.9901
Cloud	63.3552	70.9792	134.3344

TABLE VI
SMITH-WATERMAN, MEDIUM-SIZE, 3 RPIS, CLOUD: 300 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	76.4521	0.1730	76.6251
Edge - Compact	76.4952	0.0041	76.4993
Cloud	110.2217	113.4537	223.6754

that the data processing times are much longer than the data transmission times in edge computing modes, as the input size becomes larger. Hence, for a single Raspberry Pi device, its performance is worse than the Cloud.

When three Raspberry Pi devices are used, we can see that, by taking advantage of parallel processing using multiple Raspberry Pis, the data processing times in edge computing modes are shorter compared with the Cloud, which only processes data sequentially. As the Cloud spends much more time for data transmission, edge computing modes outperform the Cloud in terms of the total finishing time when two or three Raspberry Pis are used.

Figure 3 shows the performance projection for medium-size input, assuming more edge devices are used. Similar to Figure 2, we can see that with more edge devices (which leads to more iterations for the Cloud for a fair comparison), the total finishing time for the Cloud increases, whereas the edge computing remains the same performance due to parallel processing. In this medium-size input, edge computing with two or more edge devices outperforms the Cloud.

- Large-size Input:

  In this set of experiments, the two input files contain gene sequences selected from resources in [26] [27] respectively. The sizes of two input files are 4365 Bytes and 4231 Bytes, so that the problem input size is about 18.5MB (4365*4231). As shown in Tables VII, VIII, and IX, we can see that data processing times are much longer than the data transmission times, especially for edge computing modes, as the input data size becomes larger compared with small and medium-size inputs. With less powerful data processing capability, edge computing modes spend more total finishing time than the Cloud mode when one or two Raspberry Pis are used.

  We can also find that with three Raspberry Pis, the edge computing modes outperform the Cloud mode due to parallel processing of edge devices. Figure 4 shows the projected performance. We can see that starting from three edge devices, edge computing has had better performance in terms of the total finishing time.

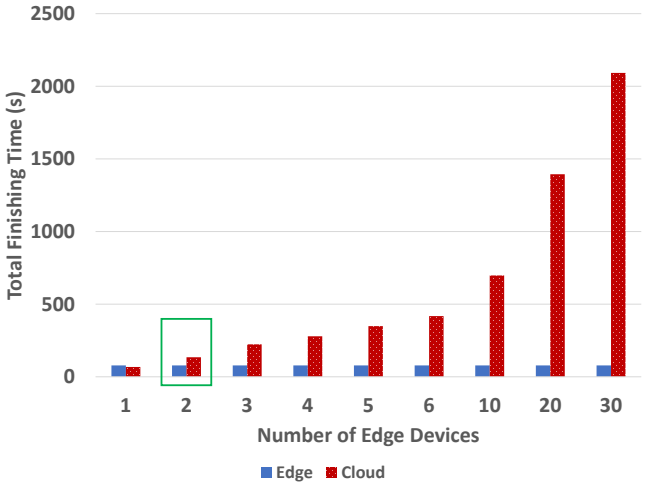

Fig. 3. Projected Performance: Smith-Waterman, Medium-size Input

TABLE VII
SMITH-WATERMAN, LARGE-SIZE, 1 RPI, CLOUD: 100 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	3075.4467	0.1731	3075.6198
Edge - Compact	2997.1033	0.0019	2997.1052
Cloud	1055.2700	37.9342	1093.2042

TABLE VIII
SMITH-WATERMAN, LARGE-SIZE, 2 RPIS, CLOUD: 200 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	3032.1150	0.1909	3032.3059
Edge - Compact	3068.3600	0.0018	3068.3618
Cloud	2088.7667	71.7193	2160.4860

TABLE IX
SMITH-WATERMAN, LARGE-SIZE, 3 RPIS, CLOUD: 300 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	3198.7700	0.1832	3198.9532
Edge - Compact	3196.5367	0.0019	3196.5386
Cloud	3135.0267	108.4443	3243.4710

Based on the above performance results in all input data sizes, we find out that edge computing modes in the Smith-Waterman gene similarity score application outperform the Cloud mode in the two following scenarios.

- When the problem input size is small enough, edge computing is better. This is because with small-size input data, the data processing time is short, so that the data transmission time is the dominant factor in the total finishing time. Hence, edge computing demonstrates its advantage by sending less amount of data to reduce the

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.

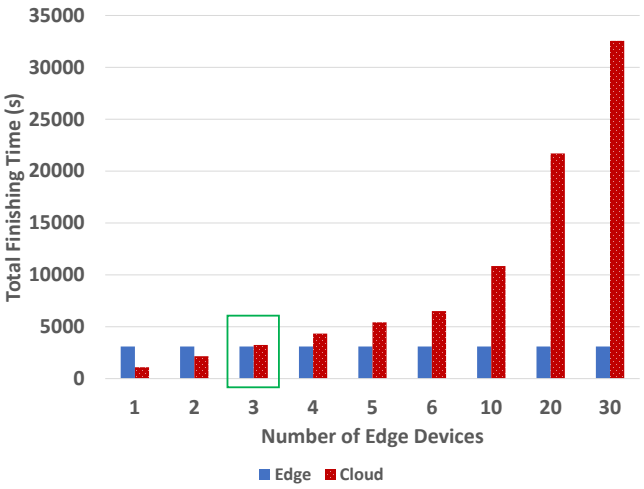

Fig. 4. Projected Performance: Smith-Waterman, Large-size Input

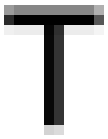

Fig. 5. Input Data for Tesseract OCR, Small-size

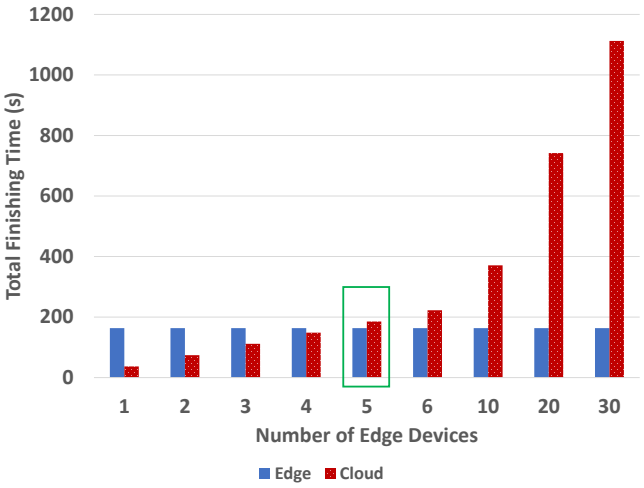

Fig. 6. Projected Performance: Tesseract OCR, Small-size Input

data transmission time.

- Edge computing outperforms the Cloud mode when there are enough edge devices which participate in the application. With more edge devices, parallel processing can show its advantage to improve the overall performance, even if a single Raspberry Pi processes data slowly.

2) Tesseract for Optical Character Recognition (OCR):

- Small-size Input:

TABLE X
TESSERACT OCR, SMALL-SIZE, 1 RPI, CLOUD: 25 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	120.2103	0.0423	120.2526
Edge - Compact	120.5097	0.0018	120.5115
Cloud	28.1984	8.7949	36.9933

TABLE XI
TESSERACT OCR, SMALL-SIZE, 2 RPIS, CLOUD: 50 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	237.8810	0.1440	238.0250
Edge - Compact	167.3300	0.0017	167.3317
Cloud	56.5934	17.3682	73.9616

TABLE XII
TESSERACT OCR, SMALL-SIZE, 3 RPIS, CLOUD: 75 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	131.7477	0.0581	131.8058
Edge - Compact	127.5387	0.0019	127.5406
Cloud	85.7468	26.0945	110.8413

In this set of experiments, the input data is an image only containing the letter "T", as shown in Figure 5. Its resolution is 20*20 pixels and the data size is 1,353 Bytes. We show the results in Tables X, XI and XII using 1, 2 and 3 Raspberry Pis respectively. We can see that for the Tesseract OCR application, the data processing time is the dominant factor in the total finishing time, even though the input data size is small. This is because Tesseract OCR application is more computing-intensive than the Smith-Waterman similarity score calculation.

Hence, the Cloud outperforms the edge computing even when three Raspberry Pis are involved, as the Cloud obviously has better data processing capability. We can also find out that with more Raspberry Pis, the performance difference between the Cloud and edge computing becomes smaller, due to parallel processing of edge devices.

Similar to the Smith-Waterman application, we show the projected performance in Figure 6, assuming there are more than three Raspberry Pi devices. We still use the average total finishing times in Tables X, XI and XII as the data for the performance projection. From Figure 6, we can see that with more Raspberry Pis, the advantage of parallel processing from edge computing is demonstrated. On the contrary, the Cloud has to process data sequentially (for a fair comparison which is the same as the Smith-Waterman application). Starting from five Rapsberry Pis, the edge computing has outperformed the Cloud.

- Medium-size Input:

In this set of experiments, the input data is an image

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.

Fig. 7. Input Data for Tesseract OCR, Medium-size

TABLE XIII
TESSERACT OCR, MEDIUM-SIZE, 1 RPI, CLOUD: 25 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	169.2657	0.0426	169.3083
Edge - Compact	169.6913	0.0018	169.6931
Cloud	45.1502	11.0544	56.2046

TABLE XIV
TESSERACT OCR, MEDIUM-SIZE, 2 RPIS, CLOUD: 50 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	193.2377	0.0536	193.2913
Edge - Compact	183.2710	0.0085	183.2795
Cloud	90.2981	22.2788	112.5769

TABLE XV
TESSERACT OCR, MEDIUM-SIZE, 3 RPIS, CLOUD: 75 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	179.4343	0.0448	179.4791
Edge - Compact	183.3900	0.0019	183.3919
Cloud	117.8573	25.9133	143.7706

of a sample vehicle license plate, as shown in Figure 7. Its resolution is 660*324 pixels and the data size is 28.1KB. Tables XIII, XIV and XV show the results for 1, 2 and 3 Raspberry Pis respectively. We can see that, similar to the result of the previous small-size input data, the data processing time here is much longer than the data transmission time for all computing modes.

We can also find out that the Cloud outperforms the edge computing in all three tables, as the Cloud processes data faster than edge devices. Also, similar to the result of the previous small-size input data, as more Raspberry Pi devices participate in the Tesseract OCR application, the difference of total finishing times between the Cloud and edge computing reduces. This is because the edge computing takes advantage of parallel processing of multiple devices.

The projected performance (assuming more than three Raspberry Pis are used) is shown in Figure 8. We can see that due to parallel processing of edge computing, when there are four or more edge devices, the edge computing outperforms the Cloud in terms of the total finishing time.

- Large-size Input:

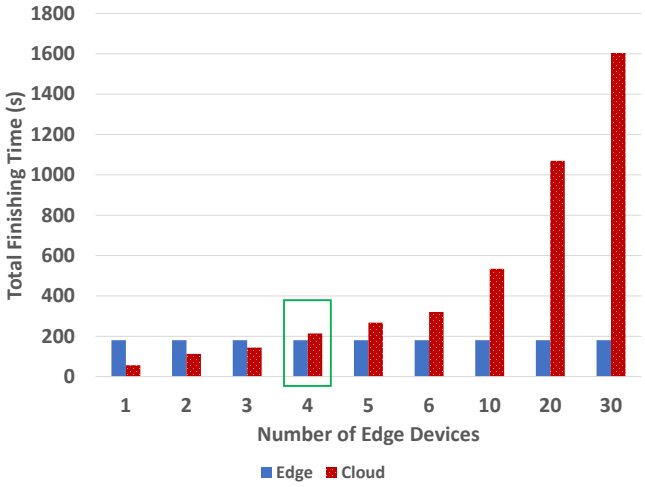

Fig. 8. Projected Performance: Tesseract OCR, Medium-size Input

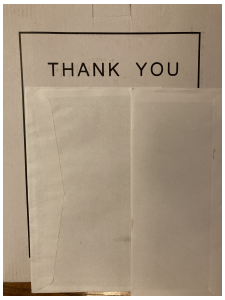

Fig. 9. Input Data for Tesseract OCR, Large-size

TABLE XVI
TESSERACT OCR, LARGE-SIZE, 1 RPI, CLOUD: 25 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	705.4970	0.0472	705.5442
Edge - Compact	705.8727	0.0018	705.8745
Cloud	181.3753	30.3810	211.7563

TABLE XVII
TESSERACT OCR, LARGE-SIZE, 2 RPIS, CLOUD: 50 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	784.6573	0.0710	784.7283
Edge - Compact	737.8230	0.0018	737.8248
Cloud	351.8570	60.4300	412.2870

TABLE XVIII
TESSERACT OCR, LARGE-SIZE, 3 RPIS, CLOUD: 75 ITERATIONS

Computing Mode	Processing Time (Sec)	Transmission Time (Sec)	Total Finishing Time (Sec)
Edge - Individual	752.3483	0.0478	752.3961
Edge - Compact	760.5120	0.0021	760.5141
Cloud	524.6440	90.0346	614.6786

In this set of experiments, the input data is an image containing the words “THANK YOU”, as shown in

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.

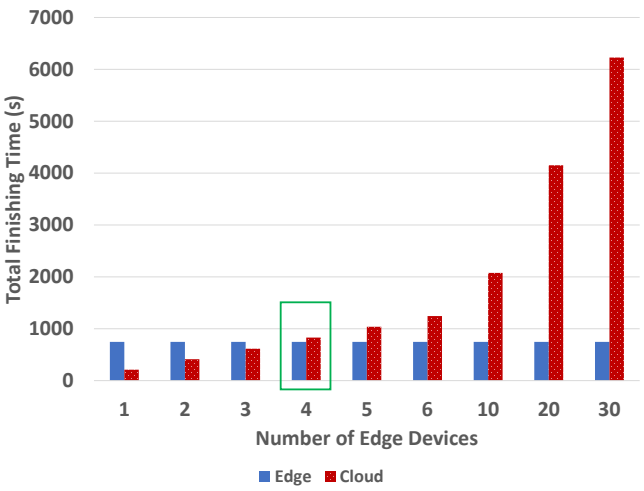

Fig. 10. Projected Performance: Tesseract OCR, Large-size Input

Figure 9. Its resolution is 3024*4032 pixels and the data size is 3.8MB. We show the results of using 1, 2 and 3 Raspberry Pis in Tables XVI, XVII and XVIII respectively. Similar to the small-size and median-size input data, the data processing time here is also the dominant factor in the total finishing time. For edge computing modes, the data transmission times are very little, as only the "THANK YOU" messages are sent.

Based on the results shown in Tables XVI, XVII and XVIII, the Cloud outperforms the edge computing modes due to its faster data processing. We can also see that, with more Raspberry Pis, the performance difference between cloud computing and edge computing modes becomes smaller. This is because the advantage of parallel processing is taken by edge computing.

Figure 10 shows the projected performance, assuming more than three edge devices are used. We can find out that with four or more Raspberry Pis, the performance of the edge computing is better than the Cloud.

Based on the above performance results in all input data sizes, we find out that with only a few Raspberry Pis, edge computing has worse performance than the Cloud in terms of the total finishing time. This is because the Tesseract OCR application requires more data processing capability. It is worth noting that edge computing can indeed significantly reduce the data transmission time. By taking advantage of parallel processing, edge computing can eventually outperform the Cloud when there are enough Raspberry Pis.

3) Key Takeaways:

Based on the above results of these two applications, we have the following important takeaways.

- In some cases, the Cloud outperforms the edge computing, while in other scenarios, edge computing has better performance. Particularly, our results verify that edge computing provides shorter total finishing time even with only one Raspberry Pi, if the data transmission time is the dominant factor when the input data size is small, as shown in the Smith-Waterman application. This is because less amount of data is transmitted in edge computing. Hence, if some applications are not computing-intensive but require short data transmission time, edge computing mode can be considered.
- On the contrary, if some applications are computing-intensive so that data processing time is the largest component in the total finishing time, cloud computing can be a good option. This conclusion is especially verified by the Tesseract application which requires more data processing power.
- Last but not least, our results and the performance projection demonstrate that parallel processing capability provided by multiple edge devices is very promising and can improve the performance. Even if a single edge server has worse performance than the Cloud for the total finishing time, multiple enough edge devices can work in parallel and will eventually outperform the Cloud. As edge servers are more affordable than Cloud servers, we can take advantage of this fact and choose edge computing for performance improvement in some cases due to parallel processing.

VII. Conclusion and Future Work

In edge computing, edge servers are usually deployed close to data sources to improve the applications' performance. Raspberry Pis nowadays have been very popular in many IoT applications. In this paper, we propose, design and implement an edge computing prototype using Raspberry Pis as edge servers. Our prototype takes advantage of local and parallel processing capabilities of Raspberry Pi devices. We demonstrate that edge computing is feasible and discuss the scenarios where edge computing has better performance. Our hands-on experiments verify that, edge computing indeed outperforms the traditional cloud computing in some cases.

In future work, we plan to explore more applications and investigate their performance using our prototype to further demonstrate the feasibility of edge computing. We also plan to deploy and implement our design on Raspberry Pi devices of the latest generation to show the performance.

References

[1] W. Shi, G. Pallis and Z. Xu, "Edge Computing [Scanning the Issue]," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1474–1481, Aug. 2019.
[2] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[3] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang and T. Zhou, "A Survey on Edge Computing Systems and Tools," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1537–1562, Aug. 2019.
[4] M. Satyanarayanan, "The Emergence of Edge Computing," in Computer, vol. 50, no. 1, pp. 30–39, Jan. 2017.
[5] N. Watkinson, F. Zaitsev, A. Shivam, M. Demirev, M. Heddes, T. Givargis, A. Nicolau, and A. Veidenbaum, "EdgeAvatar: An Edge Computing System for Building Virtual Beings," Electronics, vol. 10, no. 3, p. 229, Jan. 2021.

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.

[6] C. Li, Y. Zhang, R. Xie, X. Hao and T. Huang, "Integrating Edge Computing into Low Earth Orbit Satellite Networks: Architecture and Prototype," in IEEE Access, vol. 9, pp. 39126–39137, 2021.

[7] G. S. Pannu, J. L. L. Altet, T. Higuchi, S. Ucar, O. Altintas and F. Dressler, "Demo: Making It Real - Virtual Edge Computing in a 3D Driving Simulator," 2019 IEEE Vehicular Networking Conference (VNC), 2019, pp. 1–2.

[8] A. Mijuskovic, R. Bemthuis, A. Aldea and P. Havinga, "An Enterprise Architecture based on Cloud, Fog and Edge Computing for an Airfield Lighting Management System," in 2020 IEEE 24th International Enterprise Distributed Object Computing Workshop (EDOCW), 2020, pp. 63–73.

[9] J. Wang, J. Pan, and F. Esposito, "Elastic urban video surveillance system using edge computing," in Proceedings of the Workshop on Smart Internet of Things (SmartIoT '17), ACM, Article 7, 1—6.

[10] D. Gebbran, G. Verbič, A. C. Chapman, and S. Mhanna, "Coordination of Prosumer Agents via Distributed Optimal Power Flow: An Edge Computing Hardware Prototype," in Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '20), 2104-–2106.

[11] L. Miori, J. Sanin, and S. Helmer, "A Platform for Edge Computing Based on Raspberry Pi Clusters," in: Calì A., Wood P., Martin N., Poulovassilis A. (eds) Data Analytics, BICOD 2017, Lecture Notes in Computer Science, vol 10365, Springer, Cham.

[12] C. Pahl, S. Helmer, L. Miori, J. Sanin and B. Lee, "A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters," in 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 2016, pp. 117–124.

[13] A. Komninos, I. Simou, N. Gkorgkolis, and J. Garofalakis, "Performance of Raspberry Pi microclusters for Edge Machine Learning in Tourism," in Edge Machine Learning For Smart Iot Environments Workshop (Edging), 2019 European Conference On Ambient Intelligence (Ami2019), Rome, Italy.

[14] H. Karchalkar and P. Railkar, "Raspberry Pi Hadoop Cluster based Data Processing," in IJCA Proceedings on International Conference on Internet of Things, Next Generation Networks and Cloud Computing ICINC 2016(2):1–3, July 2016.

[15] S. Sagar, U. Choudhary and R. Dwivedi, "Smart Home Automation Using IoT and Raspberry Pi," in Proceedings of the International Conference on Innovative Computing & Communications (ICICC), 2020.

[16] S. Ferdoush and X. Li, "Wireless Sensor Network System Design using Raspberry Pi and Arduino for Environmental Monitoring Applications," in the 9th International Conference on Future Networks and Communications (FNC), 2014.

[17] J. Saffran et al, "A Low-Cost Energy-Efficient Raspberry Pi Cluster for Data Mining Algorithms," in: Desprez F. et al. (eds) Euro-Par 2016: Parallel Processing Workshops, Lecture Notes in Computer Science, vol 10104, Springer, Cham.

[18] H. Knoche and H. Eichelberger, "Using the Raspberry Pi and Docker for Replicable Performance Experiments: Experience Paper," in Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE '18), 305-–316.

[19] K. N. Myint, M. H. Zaw, and W. T. Aung, "Parallel and Distributed Computing Using MPI on Raspberry Pi Cluster," in International Journal of Future Computer and Communication vol. 9, no. 1, pp. 18–22, 2020.

[20] S.J. Cox, J.T. Cox, R.P. Boardman et al, "Iridis-pi: a low-cost, compact demonstration cluster," Cluster Computing 17, 349—358, 2014.

[21] N. Gopal, "SimpleSmithWatermanCPP," GitHub repository, Accessed on May 27, 2011. [Online]. Available: https://github.com/ngopal/SimpleSmithWatermanCPP

[22] "Smith–Waterman algorithm," in Wikipedia. [Online]. Avaiable: https://en.wikipedia.org/wiki/Smith-Waterman_algorithm

[23] zdenop, "tessdoc - API examples," [Online]. Available: https://tesseract-ocr.github.io/tessdoc/APIExample.html

[24] HIV-1 chimpanzee C455, isolate HIV-1NC, clone HIV-1NC7, from the USA, complete genome. [Online]. Available: https://www.ncbi.nlm.nih.gov/nuccore/AF049495.1

[25] HIV-1 isolate HIV100702 from Uganda pol protein (pol) gene, partial cds. [Online]. Available: https://www.ncbi.nlm.nih.gov/nucleotide/JN132241.1

[26] Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/IND/ILS01/2020, complete genome. [Online]. Available: https://www.ncbi.nlm.nih.gov/nuccore/MW559533.1

[27] Severe acute respiratory syndrome coronavirus 2 isolate SARS-CoV-2/human/USA/MA-MASPHL-01564/2021... [Online]. Available: https://www.ncbi.nlm.nih.gov/nuccore/MW560552.1/

Authorized licensed use limited to: Grand Valley State Univ. Downloaded on April 05,2024 at 15:50:56 UTC from IEEE Xplore. Restrictions apply.